NATURAL LANGUAGE INFORMATION RETRIEVAL: TREC-4 REPORT

Tomek Strzalkowski
GE Corporate Research & Development
P.O. Box 8
Schenectady, NY 12301
tomek@thuban.crd.ge.com

Jose Perez Carballo[1]
Courant Institute of Mathematical Sciences
New York University
715 Broadway, rm. 704
New York, NY 10003
carballo@cs.nyu.edu

ABSTRACT

In this paper we report on the joint GE/NYU natural language information retrieval project as related to the 4th Text Retrieval Conference (TREC-4). The main thrust of this project is to use natural language processing techniques to enhance the effectiveness of full-text document retrieval. During the course of the four TREC conferences, we have built a prototype IR system designed around a statistical full-text indexing and search backbone provided by the NIST's Prise engine. The original Prise has been modified to allow handling of multi-word phrases, differential term weighting schemes, automatic query expansion, index partitioning and rank merging, as well as dealing with complex documents. Natural language processing is used to (1) preprocess the documents in order to extract content-carrying terms, (2) discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and (3) process user's natural language requests into effective search queries. The overall architecture of the system is essentially the same as in TREC-3, as our efforts this year were directed at optimizing the performance of all components. A notable exception is the new massive query expansion module used in routing experiments, which replaces a prototype extension used in the TREC-3 system. On the other hand, it has to be noted that the character and the level of difficulty of TREC queries has changed quite significantly since the last year evaluation. TREC-4 new ad-hoc queries are far shorter, less focused, and they have a flavor of information requests (*What is the prognosis of* ...) rather than search directives typical for earlier TRECs (*The relevant document will contain* ...). This makes building of good search queries a more sensitive task than before. We thus decided to introduce only minimum number of changes to our indexing and search processes, and even roll back some of the TREC-3 extensions which dealt with longer and somewhat redundant queries (e.g., locality matching[2]). Overall, our system performed quite well as our position with respect to the best systems improved steadily since the beginning of TREC. It should be noted that the most significant gain in performance seems to occur in precision near the top of the ranking, at 5, 10, 15 and 20 documents. Indeed, our unofficial manual runs performed after TREC-4 conference show superior results in these categories, topping by a large margin the best manual scores by any system in the official evaluation.

INTRODUCTION

A typical (full-text) information retrieval (IR) task is to select documents from a database in response to a user's query, and rank these documents according to relevance. This has been usually accomplished using statistical methods (often coupled with manual encoding) that (a) select terms (words, phrases, and other units) from documents that are deemed to best represent their content, and (b) create an inverted index

[2] This turned out to be a mistake, as we explain later in this paper.

[1] Current address: School of Communication, Information and Library Studies, Rutgers University, New Brunswick, NJ 08903

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **NOV 1995**	2. REPORT TYPE	3. DATES COVERED **00-00-1995 to 00-00-1995**
4. TITLE AND SUBTITLE **Natural Language Information Retrieval: TREC-4 Report**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **GE Corporate Research & Development,PO Box 8,Schenectady,NY,12301**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **Fourth Text Retrieval Conference (TREC-4), Gaithersburg, MD, November 1-3, 1995**		

14. ABSTRACT

In this paper we report on the joint GE/NYU natural language information retrieval project as related to the 4th Text Retrieval Conference (TREC-4). The main thrust of this project is to use natural language processing techniques to enhance the effectiveness of full-text document retrieval. During the course of the four TREC conferences, we have built a prototype IR system designed around a statistical full-text indexing and search backbone provided by the NIST?s Prise engine. The original Prise has been modified to allow handling of multi-word phrases, differential term weighting schemes, automatic query expansion, index partitioning and rank merging, as well as dealing with complex documents. Natural language processing is used to (1) preprocess the documents in order to extract content-carrying terms, (2) discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and (3) process user?s natural language requests into effective search queries. The overall architecture of the system is essentially the same as in TREC-3, as our efforts this year were directed at optimizing the performance of all components. A notable exception is the new massive query expansion module used in routing experiments, which replaces prototype extension used in the TREC-3 system. On the other hand, it has to be noted that the character and the level of difficulty of TREC queries has changed quite significantly since the last year evaluation. TREC-4 new ad-hoc queries are far shorter, less focused, and they have a flavor of information requests (What is the prognosis of ...) rather than search directives typical for earlier TRECs (The relevant document will contain ...). This makes building of good search queries a more sensitive task than before. We thus decided to introduce only minimum number of changes to our indexing and search processes, and even roll back some of the TREC-3 extensions which dealt with longer and somewhat redundant queries (e.g., locality matching2). Overall, our system performed quite well as our position with respect to the best systems improved steadily since the beginning of TREC. It should be noted that the most significant gain in performance seems to occur in precision near the top of the ranking, at 5, 10, 15 and 20 documents. Indeed, our unofficial manual runs performed after TREC-4 conference show superior results in these categories, topping by a large margin the best manual scores by any system in the official evaluation.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**14**	

file (or files) that provide an easy access to documents containing these terms. A subsequent search process will attempt to match preprocessed user queries against term-based representations of documents in each case determining a degree of relevance between the two which depends upon the number and types of matching terms. Although many sophisticated search and matching methods are available, the crucial problem remains to be that of an adequate representation of content for both the documents and the queries.

In term-based representation, a document (as well as a query) is transformed into a collection of weighted terms, derived directly from the document text or indirectly through thesauri or domain maps. The representation is anchored on these terms, and thus their careful selection is critical. Since each unique term can be thought to add a new dimensionality to the representation, it is equally critical to weigh them properly against one another so that the document is placed at the correct position in the N-dimensional term space. Our goal here is to have the documents on the same topic placed close together, while those on different topics placed sufficiently apart. Unfortunately, we often do not know how to compute terms weights. The statistical weighting formulas, based on terms distribution within the database, such as *tf.idf*, are far from optimal, and the assumptions of term independence which are routinely made are false in most cases. This situation is even worse when single-word terms are intermixed with phrasal terms and the term independence becomes harder to justify.

The simplest word-based representations of content, while relatively better understood, are usually inadequate since single words are rarely specific enough for accurate discrimination, and their grouping is often accidental. A better method is to identify groups of words that create meaningful *phrases*, especially if these phrases denote important concepts in the database domain. For example, *joint venture* is an important term in the Wall Street Journal (WSJ henceforth) database, while neither *joint* nor *venture* is important by itself. In the retrieval experiments with the training TREC database, we noticed that both *joint* and *venture* were dropped from the list of terms by the system because their idf (*inverted document frequency*) weights were too low. In large databases, such as TIPSTER, the use of phrasal terms is not just desirable, it becomes necessary.

An accurate syntactic analysis is an essential prerequisite for selection of phrasal terms. Various statistical methods, e.g., based on word co-occurrences and mutual information, are prone to high error rates (sometimes as high as 50%), turning out many unwanted associations. Similarly, simplistic lexical-level parsing methods have limited potential, for example, identifying noun phrases as sequences of adjectives and nouns, adds little value to the document representation beyond what is already provided by the part-of-speech tagging. Further gains are possible if syntactic and semantic level dependencies are identified and represented. Therefore a good, fast parser may be necessary, as we have demonstrated in previous TRECs.

The challenge is to obtain ‘‘semantic’’ phrases, or ‘‘concepts’’, which would capture underlying semantic uniformity across various surface forms of expression. Syntactic structures are often reasonable indicators of content, certainly better than ‘statistical phrases’ — where words are grouped solely on the basis of physical proximity (e.g., "college junior" is not the same as "junior college") — however, the creation of compound terms makes the term matching process more complex since in addition to the usual problems of lexical meaning, one must deal with structure (e.g., "college junior" is the same as "junior in college"). In order to deal with structure, the parser’s output needs to be "normalized" or "regularized" so that complex terms with the same or closely related meanings would indeed receive matching representations. One way to regularize syntactic structures is to transform them into operator-argument form, or at least head-modifier form, as will be further explained in this paper. In effect, therefore, we aim at obtaining a semantic representation. This result has been achieved to a certain extent in our work thus far.

Do we need to parse indeed? Our recent results indicate that some of the critical *semantic* dependencies can in fact be obtained without the intermediate step of syntactic analysis, and directly from lexical-level representation of text. We have applied our noun phrase disambiguation method directly to word sequences generated using part-of-speech information, and the results were most promising. At this time we have no data how these results compare to those obtained via parsing.

No matter how we eventually arrive at the compound terms, we hope they would let us to capture more accurately the semantic content of a document. It is certainly true that the compound terms such as *South Africa*, or *advanced document processing*, when found in a document, give us a better idea about the content of such document than isolated word matches. What happens, however, if we do not find them in a document? This situation may arise for several reasons: (1) the term/concept is not there, (2) the concept is there but our system is unable to identify it, or (3) the concept is not explicitly there, but its presence can be infered using general or domain-specific knowledge.

This is certainly a serious problem, since we now attach more weight to concept matching than isolated word matching, and missing a concept can reflect more dramatically on system's recall. The inverse is also true: finding a concept where it really isn't makes an irrelevant document more likely to be highly ranked than with single-word based representation. Thus, while the rewards maybe greater, the risks are increasing as well.

One way to deal with this problem is to allow the system to fall back on partial matches and single word matches when concepts are not available, and to use query expansion techniques to supply missing terms. Unfortunately, thesaurus-based query expansion is usually quite uneffective, unless the subject domain is sufficiently narrow and the thesaurus sufficiently domain-specific. For example, the term *natural language* may be considered to subsume a term denoting a specific human language, e.g., *English*. Therefore, a query containing the former may be expected to retrieve documents containing the latter. The same can be said about *language* and *English*, unless *language* is in fact a part of the compound term *programming language* in which case the association *language* - *Fortran* is appropriate. This is a problem because (a) it is a standard practice to include both simple and compound terms in document representation, and (b) term associations have thus far been computed primarily at word level (including fixed phrases) and therefore care must be taken when such associations are used in term matching. This may prove particularly troublesome for systems that attempt term clustering in order to create "meta-terms" to be used in document representation.

In the remainder of this paper we discuss particulars of the present system and some of the observations made while processing TREC-4 data. While this description is meant to be self-contained, the reader may want to refer to previous TREC papers by this group for more information about the system.

OVERALL DESIGN

Our information retrieval system consists of a traditional statistical backbone (NIST's PRISE system; Harman and Candela, 1989) augmented with various natural language processing components that assist the system in database processing (stemming, indexing, word and phrase clustering, selectional restrictions), and translate a user's information request into an effective query. This design is a careful compromise between purely statistical non-linguistic approaches and those requiring rather accomplished (and expensive) semantic analysis of data, often referred to as 'conceptual retrieval'.

In our system the database text is first processed with a fast syntactic parser. Subsequently certain types of phrases are extracted from the parse trees and used as compound indexing terms in addition to single-word terms. The extracted phrases are statistically analyzed as syntactic contexts in order to discover a variety of similarity links between smaller subphrases and words occurring in them. A further filtering process maps these similarity links onto semantic relations (generalization, specialization, synonymy, etc.) after which they are used to transform a user's request into a search query.

The user's natural language request is also parsed, and all indexing terms occurring in it are identified. Certain highly ambiguous, usually single-word terms may be dropped, provided that they also occur as elements in some compound terms. For example, "natural" is deleted from a query already containing "natural language" because "natural" occurs in many unrelated contexts: "natural number", "natural logarithm", "natural approach", etc. At the same time, other terms may be added, namely those which are linked to some query term through admissible similarity relations. For example, "unlawful activity" is added to a query (TREC topic 055) containing the compound term "illegal activity" via a synonymy link between "illegal" and "unlawful". After the final query is constructed, the database search follows, and a ranked list of documents is returned. In TREC-4, the automatic query expansion has been limited to to routing runs, where we refined our version of massive expansion using relevenace information wrt. the training database. Query expansion via automatically generated domain map was not usd in offical ad-hoc runs.

As in TREC-3, we used a randomized index splitting mechanism which creates not one but several balanced sub-indexes. These sub-indexes can be searched independently and the results can be merged meaningfully into a single ranking.

Before we proceed to discuss the particulars of our system we would like to note that all the processing steps, those performed by the backbone system, and those performed by the natural language processing components, are fully automated, and no human intervention or manual encoding is required.

FAST PARSING WITH TTP PARSER

TTP (Tagged Text Parser) is based on the Linguistic String Grammar developed by Sager (1981). The parser currently encompasses some 400 grammar productions, but it is by no means complete. The parser's output is a regularized parse tree representation of each sentence, that is, a representation that reflects the sentence's logical predicate-argument structure. For example, logical subject and logical object are identified in both passive and active sentences, and noun phrases are organized around their head elements. The parser is equipped with a powerful skip-and-fit recovery mechanism that allows it to operate effectively in the face of ill-formed input or under a severe time pressure. When parsing the TREC-3 collection of more than 500 million words, we found that the parser's speed averaged between 0.17 and 0.26 seconds per sentence, or up to 80 words per second, on a Sun's SparcStation10. In addition, TTP has been shown to produce parse structures which are no worse than those generated by full-scale linguistic parsers when compared to hand-coded Treebank parse trees.

TTP is a full grammar parser, and initially, it attempts to generate a complete analysis for each sentence. However, unlike an ordinary parser, it has a built-in timer which regulates the amount of time allowed for parsing any one sentence. If a parse is not returned before the allotted time elapses, the parser enters the skip-and-fit mode in which it will try to "fit" the parse. While in the skip-and-fit mode, the parser will attempt to forcibly reduce incomplete constituents, possibly skipping portions of input in order to restart processing at a next unattempted constituent. In other words, the parser will favor reduction to backtracking while in the skip-and-fit mode. The result of this strategy is an approximate parse, partially fitted using top-down predictions. The fragments skipped in the first pass are not thrown out, instead they are analyzed by a simple phrasal parser that looks for noun phrases and relative clauses and then attaches the recovered material to the main parse structure. Full details of TTP parser have been described in the TREC-1 report (Strzalkowski, 1993a), as well as in other works (Strzalkowski, 1992; Strzalkowski & Scheyen, 1993).

As may be expected, the skip-and-fit strategy will only be effective if the input skipping can be performed with a degree of determinism. This means that most of the lexical level ambiguity must be removed from the input text, prior to parsing. We achieve this using a stochastic parts of speech tagger to preprocess the text (see TREC-1 report for details).

WORD SUFFIX TRIMMER

Word stemming has been an effective way of improving document recall since it reduces words to their common morphological root, thus allowing more successful matches. On the other hand, stemming tends to decrease retrieval precision, if care is not taken to prevent situations where otherwise unrelated words are reduced to the same stem. In our system we replaced a traditional morphological stemmer with a conservative dictionary-assisted suffix trimmer. [3] The suffix trimmer performs essentially two tasks: (1) it reduces inflected word forms to their root forms as specified in the dictionary, and (2) it converts nominalized verb forms (e.g., "implementation", "storage") to the root forms of corresponding verbs (i.e., "implement", "store"). This is accomplished by removing a standard suffix, e.g., "stor+age", replacing it with a standard root ending ("+e"), and checking the newly created word against the dictionary, i.e., we check whether the new root ("store") is indeed a legal word. Below is a small example of text before and after stemming.

> While serving in South Vietnam, a number of U.S. Soldiers were reported as having been exposed to the defoliant Agent Orange. The issue is veterans entitlement, or the awarding of monetary compensation and/or medical assistance for physical damages caused by Agent Orange.

> serve south vietnam number u.s. soldier expose defoliant agent orange veteran entitle award monetary compensate medical assist physical damage agent orange

Please note that proper names, such as South Vietnam and Agent Orange are identified separately through the name extraction process described below. Note also that various ‘‘stopwords’’ (e.g., prepositions, conjunctions, articles, etc.) are removed from text.

HEAD-MODIFIER STRUCTURES

Syntactic phrases extracted from TTP parse trees are head-modifier pairs. The head in such a pair is a central element of a phrase (main verb, main noun, etc.), while the modifier is one of the adjunct arguments of the head. In the TREC experiments reported here we extracted head-modifier word and fixed-phrase pairs only. While TREC databases are large enough to warrant generation of larger compounds, we were unable to verify their effectiveness in indexing, mostly because of the tight schedule.

Let us consider a specific example from the WSJ database:

> The former Soviet president has been a local hero ever since a Russian tank invaded Wisconsin.

The tagged sentence is given below, followed by the regularized parse structure generated by TTP, given in Figure 1.

[3] Dealing with prefixes is a more complicated matter, since they may have quite strong effect upon the meaning of the resulting term, e.g., *un-* usually introduces explicit negation.

The/*dt* former/*jj* Soviet/*jj* president/*nn* has/*vbz* been/*vbn* a/*dt* local/*jj* hero/*nn* ever/*rb* since/*in* a/*dt* Russian/*jj* tank/*nn* invaded/*vbd* Wisconsin/*np* ./*per*

It should be noted that the parser's output is a predicate-argument structure centered around main elements of various phrases. In Figure 1, BE is the main predicate (modified by HAVE) with 2 arguments (*subject*, *object*) and 2 adjuncts (*adv, sub_ord*). INVADE is the predicate in the subordinate clause with 2 arguments (*subject*, *object*). The subject of BE is a noun phrase with PRESIDENT as the head element, two modifiers (FORMER, SOVIET) and a determiner (THE). From this structure, we extract head-modifier pairs that become candidates for compound terms. The following types of pairs are considered: (1) a head noun and its left adjective or noun adjunct, (2) a head noun and the head of its right adjunct, (3) the main verb of a clause and the head of its object phrase, and (4) the head of the subject phrase and the main verb. These types of pairs account for most of the syntactic variants for relating two words (or simple phrases) into pairs carrying compatible semantic content. For example, the pair *retrieve+information* will be extracted from any of the following fragments: *information retrieval system; retrieval of information from databases*; and *information that can be retrieved by a user-controlled interactive search process*. In the example at hand, the following head-modifier pairs are extracted (pairs containing low-content elements, such as BE and FORMER, or names, such as WISCONSIN, will be later discarded):

```
[assert
  [[perf [HAVE]]
    [[verb [BE]]
      [subject
        [np
          [n PRESIDENT]
          [t_pos THE]
          [adj [FORMER]]
          [adj [SOVIET]]]]
      [object
        [np
          [n HERO]
          [t_pos A]
          [adj [LOCAL]]]]
      [adv EVER]
      [sub_ord
        [SINCE
          [[verb [INVADE]]
            [subject
              [np
                [n TANK]
                [t_pos A]
                [adj [RUSSIAN]]]]
            [object
              [np
                [name [WISCONSIN]]]]]]]]]
```

Figure 1. Predicate-argument parse structure.

PRESIDENT+BE, PRESIDENT+FORMER, PRESIDENT+SOVIET, BE+HERO, HERO+LOCAL,
TANK+INVADE, TANK+RUSSIAN, INVADE+WISCONSIN

We may note that the three-word phrase *former Soviet president* has been broken into two pairs *former president* and *Soviet president*, both of which denote things that are potentially quite different from what the original phrase refers to, and this fact may have potentially negative effect on retrieval precision. This is one place where a longer phrase appears more appropriate. The representation of this sentence may therefore contain the following terms (along with their inverted document frequency weights):

PRESIDENT	2.623519
SOVIET	5.416102
PRESIDENT+SOVIET	11.556747
PRESIDENT+FORMER	14.594883
HERO	7.896426
HERO+LOCAL	14.314775
INVADE	8.435012
TANK	6.848128
TANK+INVADE	17.402237
TANK+RUSSIAN	16.030809
RUSSIAN	7.383342
WISCONSIN	7.785689

While generating compound terms we took care to identify 'negative' terms, that is, those whose denotations have been explicitly excluded by negation. Even though matching of negative terms was not used in retrieval (nor did we use negative weights), we could easily prevent matching a negative term in a query against its positive counterpart in the database by removing known negative terms from queries. As an example consider the following fragment from topic 192:

> References to the cost of cleanup and number of people and equipment involved without mentioning the method are not relevant.

The corresponding compound terms are:

NOT cost cleanup
NOT number equip
NOT number people

Note that while this statement is negated, the negation is conditioned with the *without mentioning* ... phrase. Our NLP module is not able to represent such fine distinctions at this time.

NOMINAL COMPOUNDS

The notorious ambiguity of nominal compounds remains a serious difficulty in obtaining head-modifier pairs of highest accuracy. In order to cope with this, the pair extractor looks at the distribution statistics of the compound terms to decide whether the association between any two words (nouns and adjectives) in a noun phrase is both syntactically valid and semantically significant. For example, we may accept *language+natural* and *processing+language* from *natural language processing* as correct, however, *case+trading* would make a mediocre term when extracted from *insider trading case*. On the other hand, it is important to extract *trading+insider* to be able to match documents containing phrases *insider trading sanctions act* or *insider trading activity*. Phrasal terms are extracted in two phases. In the first phase, only unambiguous head-modifier pairs are generated, while all structurally ambiguous noun phrases are passed to the second phase "as is". In the second phase, the distributional statistics gathered in the first phase are used to predict the strength of alternative modifier-modified links within ambiguous phrases. For example, we may have multiple unambiguous occurrences of *insider trading*, while very few of *trading case*. At the same time, there are numerous phrases such as *insider trading case*, *insider trading legislation*, etc., where the pair *insider trading* remains stable while the other elements get changed, and significantly fewer cases where, say, *trading case* is constant and the other words change.

The disambiguation procedure is performed after the first phrase extraction pass in which all unambiguous pairs (noun+noun and noun+adjective) and all ambiguous noun phrases are extracted. Any nominal string consisting of three or more words of which at least two are nouns is deemed structurally ambiguous. In the Tipster corpus, about 80% of all ambiguous nominals were of length 3 (usually 2 nouns and an adjective), 19% were of length 4, and only 1% were of length 5 or more. The algorithm proceeds in three steps, as follows:

(1) *Assign scores* to each of the candidate pairs $x_i + x_j$ where $i > j$ from the ambiguous noun phrase $x_1 \cdots x_n$. The score assigned to a candidate pair is the sum of the scores for each occurrence of this pair in any compound nominal within the training corpus. For each occurrence, the score is maximum when the words x_i and x_j are the only words in the phrase, i.e., we have unambiguous nominal $x_j x_i$, in which case the score is 1. For longer phrases, for non-adjacent words, and for pairs anchored at words toward the left of the compound, the score decreases proportionately.

(2) For each set $X_j = \{x_i + x_j \mid \text{for } i > j\}$ of candidate pairs *rank* alternative pairs by their scores.

(3) *Disambiguate* by selecting the top choice from each set such that its score is above an empirically established global threshold, it is significantly higher than the second best choice from the set, and it is not significantly lower than the scores of pairs selected from other sets X_i.

The effectiveness of this algorithm can be measured in terms of recall (the proportion of all valid head+modifier pairs extracted from ambiguous nominals), and precision (the proportion of valid pairs among those extracted). The evaluation was done on a small sample of randomly selected phrases, and the algorithm performance was compared to manually selected correct pairs. The following numbers were recorded: recall 66% to 71%; precision 88% to 91%, depending on the size of the training sample. In terms of the total number of pairs extracted unambiguously from the parsed text (i.e., those obtained by the procedure described in the previous section), the disambiguation step recovers an additional 10% to 15% of pairs, all of which were previously thrown out as unrecoverable. A sample set of ambiguous phrases and extracted head+modifier pairs is shown in Table 1.

Ambiguous nominal	Extracted pairs
oil import fee	oil import import fee
croatian wartime cabinet	croatian cabinet wartime cabinet
national enviromental watchdog group	national group enviromental group watchdog group
current export subsidy program	current program export subsidy subsidy program
gas operating and maintaining expenses	**gas operating operating expenses maintaining expenses

Table 1. Ambiguous nominals and extracted pairs.

EXTRACTING PROPER NAMES

Proper names, of people, places, events, organizations, etc., are often critical in deciding relevance of a document. Since names are traditionally capitalized in English text, spotting them is relatively easy, most of the time. Many names are composed of more than a single word, in which case all words that make up the name are capitalized, except for prepositions and such, e.g., *The United States of America*. It is important that all names recognized in text, including those made up of multiple words, e.g., *South Africa* or *Social Security*, are represented as tokens, and not broken into single words, e.g., *South* and *Africa*, which may turn out to be different names altogether by themselves. On the other hand, we need to make sure that variants of the same name are indeed recognized as such, e.g., *U.S. President Bill Clinton* and *President Clinton*, with a degree of confidence. One simple method, which we use in our system, is to represent a compound name dually, as a compound token and as a set of single-word terms. This way, if a corresponding full name variant cannot be found in a document, its component words matches can still add to the document score. A more accurate, but arguably more expensive method would be to use a substring comparison procedure to recognize variants before matching.

In our system names are identified by the parser, and then represented as strings, e.g., *south+africa*. The name recognition procedure is extremely simple, in fact little more than the scanning of successive words labeled as proper names by the tagger (*np* and *nps* tags). Single-word names are processed just like ordinary words, except for the stemming which is not applied to them. We also made no effort to assign names to categories, e.g., people, companies, places, etc., a classification which is useful for certain types of queries (e.g., *To be relevant a document must identify a specific generic drug company*). A more advanced recognizer is planned for TREC-4 evaluation. In the TREC-3 database, compound names make up about 8% of all terms generated. A small sample of compound names extracted is listed below:

right+wing+christian+fundamentalism
u.s+constitution
gun+control+legislation
national+railroad+transportation+corporation
superfund+hazardous+waste+cleanup+programme
u.s+government
united+states
exxon+valdez
dow_corning+corporation
chairman+julius+d+winer
new+york
wall+street+journal
mcdonnell+douglas+corp+brad+beaver
soviet+georgia
rebel+leader+savimbi
plo+leader+arafat
suzuki+samurai+soft_top+4wd
honda+civic
richard+j+rosebery
mr+rosebery
international+business+machine+corp
cytomegalovirus+retinitis
ids+financial+service+analyst+g+michael+kennedy
senate+judiciary+committee
first+fidelity+bank+n.a+south+jersey
eastern+u.s
federal+national+mortgage+association
canadian+airline+international

CREATING AN INDEX

The limited amount of resources that we had available for indexing forced us to devise a method that splits the collection randomly and produces several sub-indexes. This method would allow us now to index even larger collections in reasonable times. The preliminary tests that we carried out in order to compare the performance of systems where the collection is split into N sub-indexes, for different values of N, suggest that a collection can be split into at least 7 sub-indexes without seeing any degradation in the performance. Given the results that we obtained from such tests as well as the fact that the tests were carried out using relatively small collections (about 150 Megabytes) we intend to perform more extensive testing as soon as possible.

One of the problems we had to face for TREC-1 and TREC-2 was that we did not have enough real memory to index the complete collection (category A) in a reasonable time . Even indexing only the collection for category B (550 megabytes for the ad-hoc experiments) used to take 2 weeks, or about 330 hours. This was more slow than the times that could be obtained by other versions of the PRISE system that were already available by that time. We used a slower version because we did not have then enough main memory to use the faster one. The faster version grows the word frequency tree in main memory, and it is the physical memory that matters here, not the virtual memory, since a tree larger than the size of the real memory causes so many page faults that performance becomes unacceptably slow.

The version of the PRISE system that we used for TREC-3 and TREC-4 is much faster than previous versions. According to the on-line documentation provided by NIST the old system would take about 67 hours to index 276 Megabytes of WSJ material while the new system takes less than 2 hours to index the same material. Still, we did not have enough main memory to use the new system to index the complete collection. Our solution to this problem was to split the collection into N sets of almost equal number of documents and create a separate sub-index for each set. In order to keep the N sub-indexes balanced with respect to each other (so that the term idfs are comparable across sub-indexes, for example) we split the collection randomly into N sets. This is done by assigning each document to one of the N sets selected at random. Our goal was to build N sets that would be as homogeneous as possible. At retrieval time the same query is submitted to each one of the sub-indexes and a separate list of ranked documents is obtained for each index. Since we expect idfs to be comparable across sub-indexes, it makes sense to compare the scores of documents belonging to different sub-indexes. The result of the query is then the set of documents with the highest scores chosen from the union of all lists of ranked documents.

In order to evaluate this technique we ran a series of experiments involving about 50000 records. We split that collection into N sets for several values of N (from 1 to 7) and made some measurements of parameters that we expected to be indicators of the degree of homogeneity (e.g., standard deviation of the total number of terms per index, standard deviation of the maximum idf, standard deviation of the number of unique terms, and others). As expected, these indicators showed a decreasing level of homogeneity as N grows larger. This information is summarized in Table 3.

For each value of N, we evaluated the performance of the system using a series of queries for which NIST had provided relevance judgments. For the weighting scheme we were using, and the small collection used for these preliminary experiments, we observed that the performance actually peaks at N = 4 (the average precision when N was 4 was about 7% better than when N was 1). We thought that these results were promising enough to justify the use of the technique described in order to index the complete collection but we intend to perform a much more careful and complete series of experiments as soon as the time and the resources are available. Table 4 summarizes the system's performance at various levels of index split with a subset of AP subcollection.

For TREC-4 we used 7 sub-indexes for the ad-hoc experiments (2200 Megabytes) and 5 for the routing part (1600 Megabytes). We chose these numbers because, in each case, it was the smallest number of sub-indexes that we could handle given our resources. A nice side-effect of this technique is that each index can be created in parallel on a different machine, making the total time required even shorter. The parameters of the 7-way split used in indexing the TREC-4 ad-hoc database are listed in Table 5. The reader may notice that the split is not particularly well balanced, which may be contrasted with a uniform 4-way split used in TREC-3 (cf. TREC-3 proceedings). This may have contributed to a somewhat weaker performance this year.

TERM WEIGHTING ISSUES

Finding a proper term weighting scheme is critical in term-based retrieval since the rank of a document is determined by the weights of the terms it shares with the query. One popular term weighting scheme, known as tf.idf, weights terms proportionately to their inverted document frequency scores and to their in-document frequencies (tf). The in-document

No. of indexes	Max-Mem MB	Max-idf %std	Uniq.terms Mean	Uniq.terms %std
1	81.9	0.000	921253	0.000
2	61.2	0.424	600869	1.006
3	54.7	0.902	438992	11.678
4	48.2	0.555	373249	3.095
5	46.0	0.986	314986	6.356
6	44.1	1.080	279261	7.318
7	46.8	2.432	247606	16.475

Table 3. Statistics of index splitting performed on a subset of Tipster AP88 subcollection consisting of 48,770 records (about 230 MBytes).

No. of indexes	Avg Prec %change	R-Prec %change	Recall %change
1	0.00	0.00	0.00
2	+4.04	+1.85	+1.11
3	+4.63	+0.72	+0.81
4	+7.04	+4.53	+2.59
5	+1.68	+4.08	+3.92
6	+5.68	+2.75	+4.29
7	+4.18	+4.45	+4.36

Table 4. Performance statistics for split index performed on a subset of Tipster AP88 subcollection consisting of 48,770 records (about 230 MBytes).

Index No.	Postings MB	Dict. MB	Max-idf	Records	Uniq.terms
1	60.05	31.08	17.256	78296	1426574
2	55.25	26.57	17.262	78601	1234205
3	57.69	31.10	17.206	75600	1425545
4	66.64	35.34	17.312	81400	1603649
5	60.16	30.40	17.324	82044	1394983
6	59.18	27.83	17.354	83807	1282712
7	65.47	33.33	17.419	87652	1529695

Table 5. Statistics of the 7-way split index created for ad-hoc database from Tipster Disks 2 and 3 (about 2 GBytes).

frequency factor is usually normalized by the document length, that is, it is more significant for a term to occur 5 times in a short 20-word document, than to occur 10 times in a 1000-word article.[4]

In our official TREC runs we used the normalized tf.idf weights for all terms alike: single ‘ordinary-word’ terms, proper names, as well as phrasal terms consisting of 2 or more words. Whenever phrases were included in the term set of a document, the length of this document was increased accordingly. This had the effect of decreasing tf factors for ‘regular’ single word terms.

A standard tf.idf weighting scheme (and we suspect any other uniform scheme based on frequencies) is inappropriate for mixed term sets (ordinary concepts, proper names, phrases) because:

(1) It favors terms that occur fairly frequently in a document, which supports only general-type queries (e.g., "all you know about ‘star wars’"). Such queries are not typical in TREC.

(2) It attaches low weights to infrequent, highly specific terms, such as names and phrases, whose only occurrences in a document often decide of relevance. Note that such terms cannot be reliably distinguished using their distribution in the database as the sole factor, and therefore syntactic and lexical information is required.

(3) It does not address the problem of inter-term dependencies arising when phrasal terms and their component single-word terms are all included in a document representation, i.e., *launch+satellite* and *satellite* are not independent, and it is unclear whether they should be counted as two terms.

In our post-TREC-2 experiments we considered (1) and (2) only. We changed the weighting scheme so that the phrases (but not the names which we did not distinguish in TREC-2) were more heavily weighted by their idf scores while the in-document frequency scores were replaced by logarithms multiplied by sufficiently large constants. In addition, the top N highest-idf matching terms (simple or compound) were counted more toward the document score than the remaining terms. This ‘hot-spot’ retrieval option is discussed in the next section.

Schematically, these new weights for phrasal and highly specific terms are obtained using the following formula, while weights for most of the single-word terms remain unchanged:

$$weight(T_i) = (C_1 * log(tf) + C_2 * \alpha(N,i)) * idf$$

In the above, $\alpha(N,i)$ is 1 for $i<N$ and is 0 otherwise. The $\alpha(N,i)$ factor realizes our notion of ‘‘hot spot’’ matching, where only top N matches are used in computing the document score. This creates an effect of ‘‘locality’’, somewhat similar to that achieved by passage-level retrieval (e.g., Callan, 1994). In TREC-3, where this weighing scheme was fully deployed for the first time, it proved very useful for sharpening the focus of long, frequently convoluted queries. In TREC-3 where the query length ranged from 20 to 100+ valid terms, setting N to 15 or 20 (including phrasal concepts) typically lead to a precision gain of about 20%. In TREC-4, the average query length is less than 10 terms, which we considered too short for using locality matching, and this part of the weighting scheme was in effect unused in the official runs. This turned out to be a mistake, as we rerun TREC-4 experiments after the conference, only to find out that our results improved visibly when the locality part of the weighting scheme was restored.

The table below illustrates the effect of new weighting scheme for phrasal terms using topic 101 (from TREC-2) and a relevant document (WSJ870226-0091).

Topic 101 matches WSJ870226-0091
duplicate terms not shown

TERM	*TF.IDF*	*NEW WEIGHT*
sdi	1750	1750
eris	3175	3175
star	1072	1072
wars	1670	1670
laser	1456	1456

[4] This is not always true, for example when all occurrences of a term are concentrated in a single section or a paragraph rather than spread around the article. See the following section for more discussion.

weapon	1639	1639
missile	872	872
space+base	2641	2105
interceptor	2075	2075
exoatmospheric	1879	3480
system+defense	2846	2219
reentry+vehicle	1879	3480
initiative+defense	1646	2032
system+interceptor	2526	3118
DOC RANK	30	10

Changing the weighting scheme for compound terms, along with other minor improvements (such as expanding the stopword list for topics) has lead to the overall increase of precision of 20% to 25% over our baseline results in TREC-3.

SUMMARY OF RESULTS

The bulk of the text data used in TREC-4 has been previously processed for TREC-3 (about 3.3 GBytes). Routing experiments involved some additional new text (about 500 MBytes), which we processed through our NLP module. The parameters of this process were essentially the same as in TREC-3, and an interested reader is referred to our TREC-3 paper. Two types of retrieval have been done: (1) new topics 201-250 were run in the ad-hoc mode against the Disk-2&3 database,[5] and (2) topics 3-191 (a selection of 50 topics in this range), previously used in TREC-1 to TREC-3, were run in the routing mode against the Disk-1 database plus the new data including material from Federal Register, IR Digest and Internet newsgroups. In each category 2 official runs were performed, with different set up of system's parameters. These runs were labeled *nyuge1* and *nyuge2*, for the routing runs, and *nyuge3* and *nyuge4* for adhoc runs. Both routing runs were automatic, with massive query expansion. Massive query expansion has been implemented as an automatic feedback mode using known relevance judgements for these topics with respect TREC-3 database. The adhoc runs were performed in automatic and manual modes, with nyuge3 being fully automatic, and nyuge4 using manual query expansion before search.

The purpose of the experiments we conducted this year was to find out if some techniques used by other researchers in the past (e.g., massive query expansion) would work well using our NLP techniques. The experiments we tried were the following:

(1) Routing experiment using massive expansion (the official routing run).

[5] Actually, only 49 topics were used in evaluation, since relevance judgements were unavailable for topic 201 due to an error.

(2) Ad-hoc experiment using terms added manually without previous knowledge of the documents (the official ad-hoc manual run).

(3) Ad-hoc experiment using terms selected by a user from documents found in the collection (an un-official ad-hoc semi-interactive run).

Manual ad-hoc experiments. The topics for TREC-4 were much smaller than in previous TREC's. Since our system depends on information obtained by processing the text of the topics, we decided to add text manually. The text added consisted of grammatically correct expressions that we hoped would generate phrases found in relevant documents. The extra text was added without first seeing the documents and relying only on the domain knowledge that the person adding the text might already have. No more than 2 minutes was spend to add text to any query. The results are summarized in Table 6. Notice that the ordering of the experiments with respect to precision is as we expected.

(Semi-)Interactive query expansion ad-hoc experiments. For this experiment we expanded the topics using text taken from the documents. This has been done as follows: a user submited a query and the search was run. The user then reviewed the first two pages of a number of the retrieved documents, and selected phrases from the document's text to be added to the topic. The text added was always full, grammatically correct expressions. The augmented topic were then resubmited to the system for another process/search cycle. No more than 3 cycles were used. The user spent less than 20 minutess per topic. It should be noted that this expansion did not involve the traditional relevance feedback where terms are added and reweighted based on their distribution in relevant and non-relevant sets (e.g., Roccio formula). Instead, entire phrases and sentences were added, if they appeared to be good extension of the query, which can be considered a natural elaboration of the ''off-the-top-of-your-head'' manual expansion described above. We expect that the same effect could be obtained by expanding the query using a training collection (e.g., Disk 1) different from the retrieval collection, in which case these runs would qualify as manual.

Locality runs. Following the official evaluation, we rerun all adhoc tests using the full scoring scheme that included the locality factor with *N=20*. The results turned out to be visibly better than the official runs, and the summary is given in Table 8. We also compare the locality-enhanced runs with and without phrase matching in Table 9.

Routing experiments. The relevance judgements for the routing queries wrt. the archival data were used to produce a table with 6 columns which contained the following information:

(1) query number

(2) term taken from the text of the documents

(3) rcount: number of documents that contained the term and were judged relevant.

(4) rtot: total number of documents that were judged relevant.

(5) ncount: number of documents that contained the term and that were judged not relevant.

(6) ntot total number of documents judged not relevant for the corresponding query number.

The weight of each term was computed using the following formula:

$$weight = (rcount / rtot) / (ncount / ntot)$$[6]

Summary statistics for routing runs are shown in Table 7. All runs shown in this table use massive query expansion.

In general, we can note substantial improvement in performance when phrasal terms are used, especially in ad-hoc runs. Looking back at TREC-2 and TREC-3 one may observe that these improvements appear to be tied to the length and specificity of the query: the longer the query, the more improvement from linguistic processes. This can be seen comparing the improvement over baseline for automatic adhoc runs (very short queries), for manual runs (longer queries), and for semi-interactive runs (yet longer queries). In addition, our TREC-3 results (with long and detailed queries) showed 20-25% improvement in precision attributed to NLP, as compared to 10-16% in TREC-4. At this time we are unable to explain the much smaller improvements in routing evaluations: while the massive query expansion definitely works, NLP has hard time topping these improvements.

CONCLUSIONS

We presented in some detail our natural language information retrieval system consisting of an advanced NLP module and a 'pure' statistical core engine. While many problems remain to be resolved, including the question of adequacy of term-based representation of document content, we attempted to demonstrate that the architecture described here is nonetheless viable. In particular, we demonstrated that natural language processing can now be done on a fairly large scale and that its speed and robustness has improved to the point where it can be applied to real IR problems. We suggest, with some caution until more experiments are run, that natural language processing can be very effective in creating appropriate search queries out of user's initial specifications which can be frequently imprecise or vague. An encouraging thing to note is the sharp increase of precision near the top of the ranking. This indicates a higher than average concentration of relevant documents in the first 10-20 documents retrieved, which can leverage further gains in performance via an automatic feedback process. This should be our focus in TREC-5.

At the same time it is important to keep in mind that the NLP techniques that meet our performance requirements (or at least are believed to be approaching these requirements) are still fairly unsophisticated in their ability to handle natural language text. In particular, advanced processing involving conceptual structuring, logical forms, etc., is still beyond reach, computationally. It may be assumed that these advanced techniques will prove even more effective, since they address the problem of representation-level limits; however the experimental evidence is sparse and necessarily limited to rather small scale tests.

[6] Our experiments have shown that this formula may be more effective than the traditional Roccio expansion method (see eg., Frakes & Baeza-Yates, 1992).

ACKNOWLEDGEMENTS

We would like to thank Donna Harman of NIST for making her PRISE system available to us. Will Rogers provided valuable assistance in installing updated versions of PRISE at NYU. We would also like to thank Ralph Weischedel and Constantine Papageorgiou of BBN for providing and assisting in the use of the part of speech tagger. This paper is based upon work supported by the Advanced Research Projects Agency under Contract N00014-90-J-1851 from the Office of Naval Research, under ARPA's Tipster Phase-2 Contract 94-FI57900-000, and the National Science Foundation under Grant IRI-93-02615.

REFERENCES

Broglio, John and W. Bruce Croft. 1993. ''Query Processing for Retrieval from Large Text Bases.'' Proceedings of ARPA HLT Workshop, March 21-24, Plainsboro, NJ.

Church, Kenneth Ward and Hanks, Patrick. 1990. ''Word association norms, mutual information, and lexicography.'' *Computational Linguistics*, 16(1), MIT Press, pp. 22-29.

Crouch, Carolyn J. 1988. ''A cluster-based approach to thesaurus construction.'' Proceedings of ACM SIGIR-88, pp. 309-320.

Run	abase	nyuge3	mbase	nyuge4	ibase	inlp
Queries	49	49	49	49	49	49
Tot number of docs over all queries						
Ret	46550	46997	49000	48982	49000	49000
Rel	6501	6501	6501	6501	6501	6501
RelRet	2458	2493	3410	3536	3476	3692
%chg		+1.4	+39.0	+44.0	+41.0	+50.0
Recall	Precision					
0.00	0.5296	0.6646	0.7447	0.7377	0.8103	0.8761
0.10	0.3339	0.3733	0.4650	0.5130	0.5423	0.5773
0.20	0.2586	0.2737	0.3724	0.4022	0.4077	0.4464
0.30	0.1939	0.1971	0.2997	0.3304	0.3233	0.3625
0.40	0.1585	0.1641	0.2494	0.2756	0.2740	0.3054
0.50	0.1073	0.1094	0.1714	0.1982	0.2073	0.2391
0.60	0.0831	0.0824	0.1270	0.1363	0.1417	0.1669
0.70	0.0531	0.0505	0.0913	0.0944	0.0968	0.1082
0.80	0.0253	0.0233	0.0509	0.0558	0.0462	0.0499
0.90	0.0058	0.0007	0.0141	0.0201	0.0111	0.0183
1.00	0.0000	0.0000	0.0030	0.0034	0.0006	0.0006
Average precision over all rel docs						
Avg	0.1394	0.1501	0.2082	0.2272	0.2356	0.2605
%chg		+7.7	+49.0	+63.0	+69.0	+87.0
Precision at						
5 docs	0.3755	0.4286	0.5020	0.5469	0.5837	0.6571
10 doc	0.3408	0.3918	0.4510	0.4735	0.5510	0.5898
15 doc	0.3088	0.3619	0.4082	0.4354	0.4857	0.5333
20 doc	0.2857	0.3276	0.3745	0.4163	0.4429	0.4847
30 doc	0.2483	0.2939	0.3503	0.3735	0.4014	0.4333
100 do	0.1624	0.1802	0.2451	0.2545	0.2624	0.2794
200 do	0.1211	0.1315	0.1804	0.1912	0.1869	0.2024
500 do	0.0745	0.0770	0.1069	0.1125	0.1107	0.1189
1000 d	0.0502	0.0509	0.0696	0.0722	0.0709	0.0753
R-Precision (after RelRet)						
Exact	0.1966	0.2088	0.2619	0.2780	0.2834	0.3033
%chg		+6.2	+33.0	+41.0	+44.0	+54.0

Table 6. Ad-hoc runs with queries 202-250: (1) *abase* - automatic run with statistical terms only; (2) *nyuge3* - automatic run with phrases and names; (3) *mbase* - queries manually expanded, but no phrases; (4) *nyuge4* - manual run with phrases; (5) *ibase* - semi-interactive run, no phrases; (6) *inlp* - semi-interactive with phrases.

Run	base	xbase	nyuge1	nyuge2
Queries	50	50	50	50
Tot number of docs over all queries				
Ret	50000	50000	50000	50000
Rel	6576	6576	6576	6576
RelRet	3641	4967	5078	5112
%chg		+36.0	+39.0	+40.0
Recall	(interp) Precision Averages			
0.00	0.5715	0.7420	0.7483	0.7641
0.10	0.3530	0.4898	0.5114	0.5236
0.20	0.2851	0.4220	0.4453	0.4491
0.30	0.2378	0.3614	0.3770	0.3857
0.40	0.1993	0.3145	0.3290	0.3398
0.50	0.1679	0.2730	0.2823	0.2876
0.60	0.1201	0.2285	0.2397	0.2469
0.70	0.0845	0.1701	0.1893	0.1910
0.80	0.0387	0.1263	0.1358	0.1372
0.90	0.0234	0.0652	0.0683	0.0711
1.00	0.0033	0.0067	0.0074	0.0070
Average precision over all rel docs				
Avg	0.1697	0.2715	0.2838	0.2913
%chg		+60.0	+67.0	+72.0
Precision at				
5 docs	0.3760	0.5480	0.5560	0.5680
10 docs	0.3680	0.4840	0.5000	0.5220
15 docs	0.3427	0.4680	0.4880	0.4933
20 docs	0.3240	0.4650	0.4680	0.4800
30 docs	0.3053	0.4447	0.4600	0.4680
100 docs	0.2314	0.3550	0.3658	0.3726
200 docs	0.1791	0.2790	0.2886	0.2931
500 docs	0.1142	0.1655	0.1701	0.1718
1000 docs	0.0728	0.0993	0.1016	0.1022
R-Precision (after Rel)				
Exact	0.2189	0.3100	0.3112	0.3191
%chg		+42.0	+42.0	+46.0

Table 7. Automatic routing runs with 50 queries from 3-191 range: (1) *base* - statistical terms only, no expansion; (2) *xbase* - base run with massive expansion, no phrases; (3) *nyuge1* - syntactic phrases, names, with massive query expansion of up to 500 new terms per query; (4) *nyuge2* - same as 3 but query expansion limited to 200 new terms per query.

Run	abase	aloc	mbase	mloc	ibase	iloc
Queries	49	49	49	49	49	49
Tot number of docs over all queries						
Ret	46550	47013	49000	49000	49000	49000
Rel	6501	6501	6501	6501	6501	6501
RelRet	2458	2498	3410	3545	3476	3723
%chg		+1.6	+39.0	+44.0	+41.0	+51.0
Recall	Precision					
0.00	0.5296	0.6923	0.7447	0.7525	0.8103	0.9071
0.10	0.3339	0.3702	0.4650	0.5326	0.5423	0.6039
0.20	0.2586	0.2821	0.3724	0.4138	0.4077	0.4706
0.30	0.1939	0.2207	0.2997	0.3487	0.3233	0.3936
0.40	0.1585	0.1742	0.2494	0.2941	0.2740	0.3268
0.50	0.1073	0.1345	0.1714	0.2239	0.2073	0.2578
0.60	0.0831	0.0918	0.1270	0.1513	0.1417	0.1836
0.70	0.0531	0.0565	0.0913	0.1027	0.0968	0.1178
0.80	0.0253	0.0295	0.0509	0.0641	0.0462	0.0577
0.90	0.0058	0.0006	0.0141	0.0292	0.0111	0.0227
1.00	0.0000	0.0000	0.0030	0.0045	0.0006	0.0020
Average precision over all rel docs						
Avg	0.1394	0.1592	0.2082	0.2424	0.2356	0.2767
%chg		+14.0	+49.0	+74.0	+69.0	+98.0
Precision at						
5 docs	0.3755	0.4571	0.5020	0.5592	0.5837	0.6694
10 doc	0.3408	0.3939	0.4510	0.4816	0.5510	0.6082
15 doc	0.3088	0.3687	0.4082	0.4490	0.4857	0.5633
20 doc	0.2857	0.3378	0.3745	0.4286	0.4429	0.5133
30 doc	0.2483	0.3075	0.3503	0.3925	0.4014	0.4537
100 do	0.1624	0.1927	0.2451	0.2720	0.2624	0.2978
200 do	0.1211	0.1394	0.1804	0.2051	0.1869	0.2124
500 do	0.0745	0.0798	0.1069	0.1176	0.1107	0.1240
1000 d	0.0502	0.0510	0.0696	0.0723	0.0709	0.0760
R-Precision (after RelRet)						
Exact	0.1966	0.2211	0.2619	0.2934	0.2834	0.3205
%chg		+12.0	+33.0	+49.0	+44.0	+63.0

Table 8. Ad-hoc runs with queries 202-250: (1) *abase* - automatic run with statistical terms only; (2) *aloc* - automatic run with phrases and names and locality factor set at N=20; (3) *mbase* - run with queries manually expanded, but no phrases; (4) *mloc* - manual run with phrases and locality N=20; (5) *ibase* - semi-interactive run, no phrases; (6) *iloc* - semi-interactive run with phrases, locality N=20.

	no pairs	with pairs	% change
Automatic (no locality)			
avg prec	0.1394	0.1501	7.68
prec at 10	0.3339	0.3733	11.78
Automatic (with locality)			
avg prec	0.1555	0.1592	2.38
prec at 10	0.3434	0.3702	7.80
Manual (no locality)			
avg prec	0.2082	0.2272	9.13
prec at 10	0.4650	0.5130	10.32
Manual (with locality)			
avg prec	0.2252	0.2424	7.64
prec at 10	0.4843	0.5326	9.97
Semi-Interactive (no locality)			
avg prec	0.2372	0.2626	10.71
prec at 10	0.5471	0.5843	6.80
Semi-Interactive (with locality)			
avg prec	0.2533	0.2767	9.24
prec at 10	0.5679	0.6039	6.34

Table 9. Effect of locality weighting in adhoc runs.

Frakes, William, B. and Ricardo Baeza-Yates. (eds). 1992. *Information Retrieval* Prentice-Hall, Englewood Cliffs, NJ.

Grefenstette, Gregory. 1992. ‘‘Use of Syntactic Context To Produce Term Association Lists for Text Retrieval.’’ Proceedings of SIGIR-92, Copenhagen, Denmark. pp. 89-97.

Grishman, Ralph, Lynette Hirschman, and Ngo T. Nhan. 1986. ‘‘Discovery procedures for sublanguage selectional patterns: initial experiments’’. *Computational Linguistics*, 12(3), pp. 205-215.

Grishman, Ralph and Tomek Strzalkowski. 1991. ‘‘Information Retrieval and Natural Language Processing.’’ Position paper at the workshop on Future Directions in Natural Language Processing in Information Retrieval, Chicago.

Harman, Donna. 1988. ‘‘Towards interactive query expansion.’’ Proceedings of ACM SIGIR-88, pp. 321-331.

Harman, Donna and Gerald Candela. 1989. ‘‘Retrieving Records from a Gigabyte of text on a Minicomputer Using Statistical Ranking.’’ *Journal of the American Society for Information Science,* 41(8), pp. 581-589.

Hindle, Donald. 1990. ‘‘Noun classification from predicate-argument structures.’’ Proc. 28 Meeting of the ACL, Pittsburgh, PA, pp. 268-275.

Kwok, K.L., L. Papadopoulos and Kathy Y.Y. Kwan.

1993. ‘‘Retrieval Experiments with a Large Collection using PIRCS.’’ Proceedings of TREC-1 conference, NIST special publication 500-207, pp. 153-172.

Lewis, David D. and W. Bruce Croft. 1990. ‘‘Term Clustering of Syntactic Phrases’’. Proceedings of ACM SIGIR-90, pp. 385-405.

Meteer, Marie, Richard Schwartz, and Ralph Weischedel. 1991. "Studies in Part of Speech Labeling." Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, San Mateo, CA. pp. 331-336.

Sager, Naomi. 1981. *Natural Language Information Processing.* Addison-Wesley.

Sparck Jones, Karen. 1972. ‘‘Statistical interpretation of term specificity and its application in retrieval.’’ *Journal of Documentation,* 28(1), pp. 11-20.

Sparck Jones, K. and E. O. Barber. 1971. ‘‘What makes automatic keyword classification effective?’’ *Journal of the American Society for Information Science,* May-June, pp. 166-175.

Sparck Jones, K. and J. I. Tait. 1984. ‘‘Automatic search term variant generation.’’ *Journal of Documentation,* 40(1), pp. 50-66.

Strzalkowski, Tomek and Barbara Vauthey. 1991. ‘‘Fast Text Processing for Information Retrieval.’’ Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, pp. 346-351.

Strzalkowski, Tomek and Barbara Vauthey. 1992. ‘‘Information Retrieval Using Robust Natural Language Processing.’’ Proc. of the 30th ACL Meeting, Newark, DE, June-July. pp. 104-111.

Strzalkowski, Tomek. 1992. ‘‘TTP: A Fast and Robust Parser for Natural Language.’’ Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes, France, July 1992. pp. 198-204.

Strzalkowski, Tomek. 1993. ‘‘Natural Language Processing in Large-Scale Text Retrieval Tasks.’’ Proceedings of the First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207, pp. 173-187.

Strzalkowski, Tomek. 1993. ‘‘Robust Text Processing in Automated Information Retrieval.’’ Proc. of ACL-sponsored workshop on Very Large Corpora. Ohio State Univ. Columbus, June 22.

Strzalkowski, Tomek and Jose Perez-Carballo. 1994. ‘‘Recent Developments in Natural Language Text Retrieval.’’ Proceedings of the Second Text REtrieval Conference (TREC-2), NIST Special Publication 500-215, pp. 123-136.

Strzalkowski, Tomek, Jose Perez-Carballo and Mihnea Marinescu. 1995. ‘‘Natural Language Information Retirieval: TREC-3 Report.’’ Proceedings of the Third Text REtrieval Conference (TREC-3), NIST Special Publication 500-225, pp. 39-53.

Strzalkowski, Tomek. 1995. ‘‘Natural Language Information Retrieval’’ **Information Processing and Management**, Vol. 31, No. 3, pp. 397-417. Pergamon/Elsevier.

Strzalkowski, Tomek, and Peter Scheyen. 1993. ‘‘An Evaluation of TTP Parser: a preliminary report.’’ Proceedings of International Workshop on Parsing Technologies (IWPT-93), Tilburg, Netherlands and Durbuy, Belgium, August 10-13.